



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/625,048	07/23/2003	Katherine Barabash	IL920030014US1	1091
7590	08/14/2006		EXAMINER	
Stephen C. Kaufman, Intellectual Property Law Dept. IBM Corporation P.O. Box 218 Yorktown Heights, NY 10598			SAEED, USMAAN	
			ART UNIT	PAPER NUMBER
			2166	
			DATE MAILED: 08/14/2006	

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	Application No.	Applicant(s)
	10/625,048	BARABASH ET AL.
	Examiner	Art Unit
	Usmaan Saeed	2166

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) Responsive to communication(s) filed on 12 May 2006.
- 2a) This action is FINAL.                            2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) Claim(s) 8-30 and 38-49 is/are pending in the application.
  - 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) Claim(s) \_\_\_\_\_ is/are allowed.
- 6) Claim(s) 8-30 and 38-49 is/are rejected.
- 7) Claim(s) \_\_\_\_\_ is/are objected to.
- 8) Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on 23 July 2003 is/are: a) accepted or b) objected to by the Examiner.
 

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
  - a) All    b) Some \* c) None of:
    1. Certified copies of the priority documents have been received.
    2. Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
    3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date: _____
3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date: _____	5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)
	6) <input type="checkbox"/> Other: _____

## DETAILED ACTION

### *Response to Amendment*

1. Receipt of Applicant's Amendment, filed on 5/12/2006 is acknowledged. Claims 8, 10, 11, 20, 24, 38, and 46 has been amended. New claims 47-49 have been added. Claims 1-7 and 31-37 have been cancelled.

### *Specification*

2. The examiner has withdrawn the specification objections for failing to provide proper antecedent basis for the claimed subject matter.

### *Claim Rejections - 35 USC § 103*

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were

made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 12-15, 20-23, 42, 44, 46, and 48 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Printezis et al.** (NPL “A Generational Mostly-Concurrent Garbage Collector”).

With respect to claim 12, **Printezis** teaches a **method for collecting garbage in a computing environment, the method comprising:**

“**a) tracing a root object to any of its reachable objects in a population of objects**” as record all objects directly reachable from the *roots* (globals, stacks, registers) of the system (**Printezis** Page 4 “**Mostly Concurrent Collection (Initial marking pause)**”).

“**b) marking any of said objects referred to in step a)**” as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects (**Printezis** Page 4 “**Mostly Concurrent Collection (Concurrent marking phase)**”).

“**c) unmarking a marked card comprising any of said objects**” as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the

heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack. Figure 1a shows the heap halfway through the concurrent marking phase. Objects **b**, **c**, and **e** have been marked. At this point, the mutator performs two updates: object **g** drops its reference to **d**, and object **b** has its reference field, which pointed to **c**, overwritten with a reference to **d**. The result of these updates is illustrated in Figure 1b. Also note that the updates caused pages 1 and 3 to be dirtied (**Printezis** Page 5 “A Concrete Example” & Figure 1). The examiner interprets a page as a card since the page comprises the objects. The examiner also interprets dirtied page to be marked and clean pages as unmarked.

- “d) tracing any marked object on said unmarked card to an unmarked referent object of said marked object**
- e) marking said unmarked referent object**
- f) tracing said referent object marked in step e) to any of its reachable objects**
- g) marking any of said objects referred to in step f)**
- h) tracing any unmarked root object referent to any of its reachable objects**
- i) marking any of said objects referred to in step h)”** as complete the marking phase by marking from the roots, considering modified reference fields in marked objects as additional roots. Since such fields contain the only references that the concurrent marking phase may not have observed, this ensures that the final transitive closure includes all objects reachable at the start of the final marking phase (**Printezis**

Page 4 “Final marking pause” & Figure 1). These lines and figure 1 teaches us that there are multiple steps of tracing and marking objects.

“j) performing any of steps c)-g)” as it may also include some objects that became unreachable after they were marked. These will be collected during the next garbage collection cycle (Printezis Page 4 “Final marking pause”).

“k) designating any unmarked object in said population of objects as available for reallocation” as every such path consists either entirely of unmarked objects allocated during marking, or contains at least one marked object (Printezis Page 16 “4.8 Concurrency Issues”). These lines teach that the unmarked objects are allocated during marking.

“wherein any of steps a)-g) are performed upon said population of objects concurrently with the operation of a mutator upon said population of objects within said computing environment” as a garbage collection algorithm that has been designed to serve as the oldest generation of a generational memory system. It attempts to decrease the worst-case garbage collection pause time, while taking advantage of the benefits of a generational system. It is an adaptation of the “mostly parallel” algorithm of Boehm *et al.* [6]. It usually operates concurrently with the mutator, only occasionally suspending the mutator for short periods (Printezis Page 2 “1. Introduction”) and “wherein any of steps h)-k) are performed upon said population of objects while no mutator operates upon said population of objects within said computing environment” as there are some ways in which our mostly-concurrent collector has been optimized or modified to work as the older generation in a

generational collector. First, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation. (The remainder is “direct” allocation by the mutator in the older generation, which usually occurs only for objects too large to be allocated in the young generation). Promotion occurs while mutator threads and the concurrent garbage collector thread are suspended, which simplifies matters. We take advantage of this simplification by supporting a *linear allocation mode* during young-generation collection (**Printezis** Page 13 “**4.6 Interaction with Young-Generation Collection**”). The examiner interprets the steps a)-g) as oldest and older generation collection and steps h)-k) as young generation collection. These lines teach us that the mutators are suspended and do not operate for the young generation collection.

**Printezis** teaches the elements of claim 12 as noted above but does not explicitly disclose the step of “**wherein either of steps a) and f) are performed for a given object only if the card to which the object belongs is not marked.**”

However, **Printezis** discloses “**wherein either of steps a) and f) are performed for a given object only if the card to which the object belongs is not marked**” as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object a is marked live, since it is reachable from a thread stack (**Printezis** Page 5 “**A Concrete Example**” & Figure 1). The examiner interprets clean pages as unmarked and dirty pages as

marked. In figure 1a pages are unmarked/clean and objects are traced on these cards/pages.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Printezis** teaching would have provided a faster tracing by only tracing objects if the card to which they belong are not marked.

Claim 42 and 46 is essentially the same as claim 12 except it sets forth the claimed invention as a system and a computer program and is rejected for the same reasons as applied hereinabove.

With respect to claim 13, **Printezis** teaches “**marking said card if said mutator modifies an object pointer of an object in said card**” as the base generational system, a young-generation collection scans all dirty old-space cards, searching for pointers into the young generation. If none are found, there is no need to scan this card in the next collection, so the card is marked as clean. Before a young-generation collection cleans a dirty card, the information that the card has been modified must be recorded for the mostly-concurrent collector (**Printezis** Page 9 “**4.2 Using the Card Table**”). In figure one when an object pointer is modified the page/card is marked/dirtied.

Claims 21 is same as claim 13 and is rejected for the same reasons as applied hereinabove.

With respect to claim 14, **Printezis** teaches “**wherein any of steps a)-g) are performed concurrently**” as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects. This closure is *not* guaranteed to contain all objects reachable at the end of marking, since concurrent updates of reference fields by the mutator may have prevented the marking phase from reaching some live objects (**Printezis** Page 4 “**Mostly Concurrent Collection (Concurrent marking phase)**”).

Claims 22 is same as claim 14 and is rejected for the same reasons as applied hereinabove.

With respect to claim 15, **Printezis** teaches “**wherein any of steps h)-j) are performed concurrently**” as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects. This closure is *not* guaranteed to contain all objects reachable at the end of marking, since concurrent updates of reference fields by the mutator may have prevented the marking phase from reaching some live objects (**Printezis** Page 4 “**Mostly Concurrent Collection (Concurrent marking phase)**”).

Claims 23 is same as claim 15 and is rejected for the same reasons as applied hereinabove.

With respect to claim 20, **Printezis teaches a method for collecting garbage in a computing environment, the method comprising:**

**“a) tracing a root object to any of its reachable objects in a population of objects”** as record all objects directly reachable from the *roots* (globals, stacks, registers) of the system (**Printezis Page 4 “Mostly Concurrent Collection (Initial marking pause)”**).

**“b) marking any of said objects referred to in step a”** as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects (**Printezis Page 4 “Mostly Concurrent Collection (Concurrent marking phase)”**).

**“c) unmarking a marked card comprising any of said objects”** as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack. Figure 1a shows the heap halfway through the concurrent marking phase. Objects **b**, **c**, and **e** have been marked. At this point, the mutator performs two updates: object **g** drops its reference to **d**, and object **b** has its reference field, which pointed to **c**, overwritten with a reference to **d**. The result of these updates is illustrated in Figure 1b. Also note that the updates caused pages 1 and 3 to

be dirtied (**Printezis** Page 5 “**A Concrete Example**” & Figure 1). The examiner interprets a page as a card since the page comprises the objects. The examiner also interprets dirtied page to be marked and clean pages as unmarked.

**“d) tracing any marked object on said unmarked card to an unmarked referent object of said marked object**

**e) marking said unmarked referent object**

**f) tracing said referent object marked in step e) to any of its reachable objects**

**g) marking any of said objects referred to in step f)**

**h) tracing any unmarked root object referent to any of its reachable objects**

**i) marking any of said objects referred to in step h)”** as complete the marking phase by marking from the roots, considering modified reference fields in marked objects as additional roots. Since such fields contain the only references that the concurrent marking phase may not have observed, this ensures that the final transitive closure includes all objects reachable at the start of the final marking phase (**Printezis** Page 4 “**Final marking pause**” & Figure 1). These lines and figure 1 teaches us that there are multiple steps of tracing and marking objects.

**“j) performing any of steps c)-g)”** as it may also include some objects that became unreachable after they were marked. These will be collected during the next garbage collection cycle (**Printezis** Page 4 “**Final marking pause**”).

**“k) designating any unmarked object in said population of objects as available for reallocation”** as every such path consists either entirely of unmarked

objects allocated during marking, or contains at least one marked object (**Printezis Page 16 “4.8 Concurrency Issues”**). These lines teach that the unmarked objects are allocated during marking.

**“wherein any of steps a)-g) are performed upon said population of objects concurrently with the operation of a mutator upon said population of objects within said computing environment”** as a garbage collection algorithm that has been designed to serve as the oldest generation of a generational memory system. It attempts to decrease the worst-case garbage collection pause time, while taking advantage of the benefits of a generational system. It is an adaptation of the “mostly parallel” algorithm of Boehm *et al.* [6]. It usually operates concurrently with the mutator, only occasionally suspending the mutator for short periods (**Printezis Page 2 “1. Introduction”**) and **“wherein any of steps h)-k) are performed upon said population of objects while no mutator operates upon said population of objects within said computing environment”** as there are some ways in which our mostly-concurrent collector has been optimized or modified to work as the older generation in a generational collector. First, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation. (The remainder is “direct” allocation by the mutator in the older generation, which usually occurs only for objects too large to be allocated in the young generation). Promotion occurs while mutator threads and the concurrent garbage collector thread are suspended, which simplifies matters. We take advantage of this simplification by supporting a *linear allocation mode* during young-generation collection (**Printezis Page**

13 “4.6 Interaction with Young-Generation Collection”). The examiner interprets the steps a)-g) as oldest and older generation collection and steps h)-k) as young generation collection. These lines teach us that the mutators are suspended and do not operate for the young generation collection.

**Printezis** teaches the elements of claim 20 as noted above but does not explicitly disclose the step of “**l) at any time relative to performing any of steps a)-g), periodically unmarking any marked card that does not contain at least one of said marked objects.**”

However, **Printezis** discloses “**l) at any time relative to performing any of steps a)-g), periodically unmarking any marked card that does not contain at least one of said marked objects**” as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object a is marked live, since it is reachable from a thread stack. A concurrent sweeping phase follows, and will reclaim the unmarked object f (**Printezis** Page 5 “A Concrete Example” & Figure 1). The examiner interprets clean pages as unmarked and dirty pages as marked. In figure 1d page 3 is unmarked from being marked.

Further **Printezis** discloses dirty cards are changed to precleaned, which are considered dirty by generational collection, but considered clean in the final mark phase (**Printezis** Page 17, 4.9 Concurrent Precleaning) and a young generation collection

cleans a dirty card, the information that the card has been modified must be recorded for the mostly-concurrent collector (**Printezis** Page 9, 4.2 Using the card table).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Printezis** teaching would have provided a faster tracing by unmarking any marked card that does not contain at least one of said marked objects.

Claim 44 is essentially the same as claim 20 except it sets forth the claimed invention as a system and is rejected for the same reasons as applied hereinabove.

Claim 48 is same as claim 20 and is rejected for same reasons as applied hereinabove.

4. Claims 8, 10, 16-19, 24-27, 29, 38, 40, 43, 45, 47, and 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Printezis et al.** (NPL "A Generational Mostly-Concurrent Garbage Collector") as applied to claims 12-15, 20-23, 42, 44, 46, and 48 above, in view of **Kolodner et al.** (**Kolodner** hereinafter) (U.S. PGPub No. 2001/0000821).

With respect to claim 8, **Printezis** teaches "**a method according to claim 1 and further comprising: designating any of said objects as "new""**" as care must be taken not to deallocate newly allocated objects. This can be accomplished by allocating objects "live" (i.e., marked), at least during this phase (**Printezis** Page 4 "**Mostly**

**Concurrent Collection (Concurrent sweeping phase)"**). Examiner interprets "live" as "new".

**Printezis** teaches the elements of claim 8 as noted above but does not explicitly disclose the step of "**deferring the tracing of said "new" objects during plurality of cycles.**"

However, **Kolodner** discloses, "**deferring the tracing of said "new" objects during plurality of cycles**" as these new objects may die quite young, but since they are created black (or changed from white to black), they will not be collected until the next full collection, despite the fact that they die young and could have been collected and reclaimed much earlier (**Kolodner** Paragraph 0136).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Kolodner's** teaching would have allowed **Printezis** to increase the collection and reallocation time since there is no risk of "missing" any live object during trace because the second check is obviated since all live objects are going to become old and accordingly no pointer can become inter-generational (i.e. pointing from old to young).

Claim 38 is essentially the same as claim 8 except it sets forth the claimed invention as a system and is rejected for the same reasons as applied hereinabove.

With respect to claim 10, **Printezis** teaches a method according to claim 8 and further comprising:

**“periodically unmarking any marked card containing only “new” objects”**

as figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack (**Printezis Page 5 “A Concrete Example” & Figure 1**). The examiner interprets clean pages as unmarked and dirty pages as marked. In figure 1a page 3 is unmarked/clean and have unmarked objects on this card/page.

**“removing said “new” objects’ “new” designation”** as first, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation (**Printezis Page 13 “Interaction with Young-Generation Collection”**). Only the young generations are new but after the promotion to older generation they are not new anymore.

Claims 29 and 40 are same as claim 10 except claim 40 sets forth the claimed invention as a system and are rejected for the same reasons as applied hereinabove.

With respect to claim 16, **Printezis teaches a method for collecting garbage in a computing environment, the method comprising:**

**“a) tracing a root object to any of its reachable objects in a population of objects”** as record all objects directly reachable from the *roots* (globals, stacks,

registers) of the system (**Printezis Page 4 “Mostly Concurrent Collection (Initial marking pause)”**).

**“b) marking any of said objects referred to in step a)”** as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects (**Printezis Page 4 “Mostly Concurrent Collection (Concurrent marking phase)”**).

**“c) unmarking a marked card comprising any of said objects”** as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack. Figure 1a shows the heap halfway through the concurrent marking phase. Objects **b**, **c**, and **e** have been marked. At this point, the mutator performs two updates: object **g** drops its reference to **d**, and object **b** has its reference field, which pointed to **c**, overwritten with a reference to **d**. The result of these updates is illustrated in Figure 1b. Also note that the updates caused pages 1 and 3 to be dirtied (**Printezis Page 5 “A Concrete Example” & Figure 1**). The examiner interprets a page as a card since the page comprises the objects. The examiner also interprets dirtied page to be marked and clean pages as unmarked.

**“d) tracing any marked object on said unmarked card to an unmarked referent object of said marked object**

**e) marking said unmarked referent object**

**f) tracing said referent object marked in step e) to any of its reachable objects**

**g) marking any of said objects referred to in step f)**

**h) tracing any unmarked root object referent to any of its reachable objects**

**i) marking any of said objects referred to in step h)"** as complete the marking phase by marking from the roots, considering modified reference fields in marked objects as additional roots. Since such fields contain the only references that the concurrent marking phase may not have observed, this ensures that the final transitive closure includes all objects reachable at the start of the final marking phase (**Printezis Page 4 “Final marking pause” & Figure 1**). These lines and figure 1 teaches us that there are multiple steps of tracing and marking objects.

**“j) performing any of steps c)-g)”** as it may also include some objects that became unreachable after they were marked. These will be collected during the next garbage collection cycle (**Printezis Page 4 “Final marking pause”**).

**“k) designating any unmarked object in said population of objects as available for reallocation”** as every such path consists either entirely of unmarked objects allocated during marking, or contains at least one marked object (**Printezis Page 16 “4.8 Concurrency Issues”**). These lines teach that the unmarked objects are allocated during marking.

**“wherein prior to said unmarking step c) said card is marked only if there is at least one marked object already on said card”** as figure 1a shows the heap halfway through the concurrent marking phase. Objects b, c, and e have been marked.

At this point, the mutator performs two updates: object g drops its reference to d, and object b has its reference field, which pointed to c, overwritten with a reference to d. The result of these updates is illustrated in Figure 1b. Also note that the updates caused pages 1 and 3 to be dirtied (Printezis Page 5 “A Concrete Example” & Figure 1). The examiner interprets clean pages as unmarked and dirty pages as marked. Page 1 in figure 1b and page 1&3 in figure 1c has marked objects and the card itself is also marked/dirtied. **And “wherein any of steps a)-g) are performed upon said population of objects concurrently with the operation of a mutator upon said population of objects within said computing environment”** as a garbage collection algorithm that has been designed to serve as the oldest generation of a generational memory system. It attempts to decrease the worst-case garbage collection pause time, while taking advantage of the benefits of a generational system. It is an adaptation of the “mostly parallel” algorithm of Boehm *et al.* [6]. It usually operates concurrently with the mutator, only occasionally suspending the mutator for short periods (**Printezis Page 2 “1. Introduction”**) and **“wherein any of steps h)-k) are performed upon said population of objects while no mutator operates upon said population of objects within said computing environment”** as there are some ways in which our mostly-concurrent collector has been optimized or modified to work as the older generation in a generational collector. First, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation. (The remainder is “direct” allocation by the mutator in the older generation, which usually occurs only for objects too large to be allocated in the young generation).

Promotion occurs while mutator threads and the concurrent garbage collector thread are suspended, which simplifies matters. We take advantage of this simplification by supporting a *linear allocation mode* during young-generation collection (**Printezis** Page 13 “**4.6 Interaction with Young-Generation Collection**”). The examiner interprets the steps a)-g) as oldest and older generation collection and steps h)-k) as young generation collection. These lines teach us that the mutators are suspended and do not operate for the young generation collection.

**Printezis** teaches elements of claim 16 as noted above but does not explicitly teach “**said card is marked only if there is at least one marked object already on said card.**”

However, **Kolodner** discloses “**said card is marked only if there is at least one marked object already on said card**” as a dirty bit for each card that contains a dirty object and in addition one dirty bit per object (**Kolodner** Paragraph 0111). Therefore the card is marked/dirtied only if it contains a dirty/marketed object.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Kolodner’s** teaching would have allowed **Printezis** to save time for the collector, since the collector can tell exactly which object in a dirty card is actually dirty and does not have to trace all objects in the card.

Claim 43 is essentially the same as claim 16 except it sets forth the claimed invention as a system and is rejected for the same reasons as applied hereinabove.

Claim 47 is same as claim 16 and is rejected for same reasons as applied hereinabove.

Claims 17-19 are same as claims 13-15 and are rejected for the same reasons as applied hereinabove.

With respect to claim 24, **Printezis** teaches a **method for collecting garbage in a computing environment, the method comprising:**

**“a) tracing a root object to any of its reachable objects in a population of objects”** as record all objects directly reachable from the *roots* (globals, stacks, registers) of the system (**Printezis** Page 4 “**Mostly Concurrent Collection (Initial marking pause)**”).

**“b) marking any of said objects referred to in step a)”** as at the same time, initiate a concurrent marking phase, which marks a transitive closure of reachable objects (**Printezis** Page 4 “**Mostly Concurrent Collection (Concurrent marking phase)**”).

**“c) unmarking a marked card comprising any of said objects”** as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack. Figure 1a shows the heap halfway through the concurrent marking phase. Objects **b**, **c**, and **e** have been marked. At this point, the

mutator performs two updates: object **g** drops its reference to **d**, and object **b** has its reference field, which pointed to **c**, overwritten with a reference to **d**. The result of these updates is illustrated in Figure 1b. Also note that the updates caused pages 1 and 3 to be dirtied (**Printezis** Page 5 “A Concrete Example” & Figure 1). The examiner interprets a page as a card since the page comprises the objects. The examiner also interprets dirtied page to be marked and clean pages as unmarked.

**“d) tracing any marked object on said unmarked card to an unmarked referent object of said marked object**

**e) marking said unmarked referent object**

**f) tracing said referent object marked in step e) to any of its reachable objects**

**g) marking any of said objects referred to in step f)**

**h) tracing any unmarked root object referent to any of its reachable objects**

**i) marking any of said objects referred to in step h)”** as complete the marking phase by marking from the roots, considering modified reference fields in marked objects as additional roots. Since such fields contain the only references that the concurrent marking phase may not have observed, this ensures that the final transitive closure includes all objects reachable at the start of the final marking phase (**Printezis** Page 4 “Final marking pause” & Figure 1). These lines and figure 1 teaches us that there are multiple steps of tracing and marking objects.

**“j) performing any of steps c)-g)”** as it may also include some objects that became unreachable after they were marked. These will be collected during the next garbage collection cycle (**Printezis Page 4 “Final marking pause”**).

**“k) designating any unmarked object in said population of objects as available for reallocation”** as every such path consists either entirely of unmarked objects allocated during marking, or contains at least one marked object (**Printezis Page 16 “4.8 Concurrency Issues”**). These lines teach that the unmarked objects are allocated during marking.

**“l) at any time relative to performing any steps a)-g): designating any of said objects as “new””** as care must be taken not to deallocate newly allocated objects. This can be accomplished by allocating objects “live” (i.e., marked), at least during this phase (**Printezis Page 4 “Mostly Concurrent Collection (Concurrent sweeping phase”**). Examiner interprets “live” as “new”.

**“wherein any of steps a)-g) are performed upon said population of objects concurrently with the operation of a mutator upon said population of objects within said computing environment”** as a garbage collection algorithm that has been designed to serve as the oldest generation of a generational memory system. It attempts to decrease the worst-case garbage collection pause time, while taking advantage of the benefits of a generational system. It is an adaptation of the “mostly parallel” algorithm of Boehm *et al.* [6]. It usually operates concurrently with the mutator, only occasionally suspending the mutator for short periods (**Printezis Page 2 “1. Introduction”**) and **“wherein any of steps h)-k) are performed upon said**

**population of objects while no mutator operates upon said population of objects within said computing environment**" as there are some ways in which our mostly-concurrent collector has been optimized or modified to work as the older generation in a generational collector. First, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation. (The remainder is "direct" allocation by the mutator in the older generation, which usually occurs only for objects too large to be allocated in the young generation). Promotion occurs while mutator threads and the concurrent garbage collector thread are suspended, which simplifies matters. We take advantage of this simplification by supporting a *linear allocation mode* during young-generation collection (**Printezis** Page 13 "4.6 Interaction with Young-Generation Collection"). The examiner interprets the steps a)-g) as oldest and older generation collection and steps h)-k) as young generation collection. These lines teach us that the mutators are suspended and do not operate for the young generation collection.

**Printezis** teaches the elements of claim 24 as noted above but does not explicitly disclose the step of "**deferring the tracing of said "new" objects.**"

However, **Kolodner** discloses, "**deferring the tracing of said "new" objects**" as these new objects may die quite young, but since they are created black (or changed from white to black), they will not be collected until the next full collection, despite the fact that they die young and could have been collected and reclaimed much earlier (**Kolodner** Paragraph 0136).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Kolodner's** teaching would have allowed **Printezis** to increase the collection and reallocation time since there is no risk of "missing" any live object during trace because the second check is obviated since all live objects are going to become old and accordingly no pointer can become inter-generational (i.e. pointing from old to young).

Claim 45 is essentially the same as claim 24 except it sets forth the claimed invention as a system and is rejected for the same reasons as applied hereinabove.

Claim 49 is same as claim 24 and is rejected for same reasons as applied hereinabove.

Claims 25-27 are same as claims 13-15 and are rejected for the same reasons as applied hereinabove.

5. Claims 9, 11, 28, 30, 39, and 41 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Printezis et al.** (NPL "A Generational Mostly-Concurrent Garbage Collector") in view of **Kolodner et al.** (U.S. PGPub No. 2001/0000821) further in view of **Johnson et al. (Johnson hereinafter)** (U.S. Patent No. 5,948,113).

With respect to claim 9, **Printezis** teaches "**a method according to claim 8 wherein said designating as "new" step is performed**" as care must be taken not to

deallocate newly allocated objects. This can be accomplished by allocating objects “live” (i.e., marked), at least during this phase (**Printezis Page 4 “Mostly Concurrent Collection (Concurrent sweeping phase)”**).

**Printezis** teaches the elements of claim 9 as noted above but does not explicitly disclose the step of **“if said object is part of an allocation cache from which objects are currently being allocated.”**

However, **Johnson** discloses **“if said object is part of an allocation cache from which objects are currently being allocated”** as the global object manager either allocates the new object directly from memory or attempts to reuse a previously allocated object in a cache of available objects as the new object (**Johnson Abstract**).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Johnson’s** teaching would have allowed **Printezis** to advantageously minimize and avoid unnecessary memory fragmentation by re-use of allocated objects.

Claims 28 and 39 are same as claim 9 except claim 39 sets forth the claimed invention as a system and are rejected for the same reasons as applied hereinabove.

With respect to claim 11, **Printezis** teaches **“a method according to claim 10 wherein said periodically unmarking and removing steps are performed”** as figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not

illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack (**Printezis** Page 5 “**A Concrete Example**” & Figure 1). First, we recognize that, for most programs, a large majority of allocation in the older generation will be done via promotion from the young generation (**Printezis** Page 13 “**Interaction with Young-Generation Collection**”). The examiner interprets clean pages as unmarked and dirty pages as marked. In figure 1a page 3 is unmarked/clean and have unmarked objects on this card/page. Only the young generations are new but after the promotion to older generation they are not new anymore.

**Printezis** teaches the elements of claim 11 as noted above but does not explicitly disclose the step of “**if said object is part of an allocation cache from which objects are not currently being allocated**.”

However, **Johnson** discloses “**if said object is part of an allocation cache from which objects are not currently being allocated**” as if a re-use condition is met, the object is kept in the cache as an available object already allocated from memory. However, if the re-use condition is not met, the object is de-allocated from memory (**Johnson** Abstract). The de-allocated object is interpreted as currently not being allocated from cache/memory.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Johnson’s** teaching would have allowed **Printezis** to advantageously minimize and avoid unnecessary memory fragmentation by re-use of allocated objects

Claims 30 and 41 are same as claim 11 except claim 41 sets forth the claimed invention as a system and are rejected for the same reasons as applied hereinabove.

***Response to Arguments***

6. Applicant's arguments with respect to claims 12, 16, and 20 have been considered but are moot in view of the new ground(s) of rejection.

Regarding claim 12, applicant argues that **Printezis** does not teach “**tracing an object only if the object is on an unmarked card.**”

In response to the preceding argument, Examiner respectfully submits that **Printezis** does not explicitly disclose the step of “**tracing an object only if the object is on an unmarked card.**”

However, **Printezis** discloses “**tracing an object only if the object is on an unmarked card**” as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object **a** is marked live, since it is reachable from a thread stack (**Printezis** Page 5 “**A Concrete Example**” & Figure 1). The examiner interprets clean pages as unmarked and dirty pages as marked. In figure 1a pages are unmarked/clean and objects are traced on these cards/pages.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Printezis** teaching would have provided a faster tracing by only tracing objects if the card to which they belong are not marked.

Regarding claim 16, applicant argues that **Printezis** does not teach “**said card is marked only if there is at least one marked object already on said card.**”

In response to the preceding argument, Examiner respectfully submits that **Printezis** does not explicitly disclose the step of “**said card is marked only if there is at least one marked object already on said card.**”

However, **Kolodner** discloses “**said card is marked only if there is at least one marked object already on said card**” as a dirty bit for each card that contains a dirty object and in addition one dirty bit per object (**Kolodner** Paragraph 0111). Therefore the card is marked/dirtied only if it contains a dirty-marked object.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Kolodner’s** teaching would have allowed **Printezis** to save time for the collector, since the collector can tell exactly which object in a dirty card is actually dirty and does not have to trace all objects in the card.

Regarding claim 20, applicant argues that **Printezis** does not teach “periodically unmarking any marked card that does not contain at least one of said marked objects.”

In response to the preceding argument, Examiner respectfully submits that **Printezis** does not explicitly disclose the step of “periodically unmarking any marked card that does not contain at least one of said marked objects.”

However, **Printezis** discloses “periodically unmarking any marked card that does not contain at least one of said marked objects” as Figure 1 illustrates the operation of the mostly-concurrent algorithm. In this simple example, the heap contains 7 objects and is split into 4 pages. During the initial marking pause (not illustrated), all 4 pages are marked as clean and object a is marked live, since it is reachable from a thread stack. A concurrent sweeping phase follows, and will reclaim the unmarked object f (**Printezis** Page 5 “A Concrete Example” & Figure 1). The examiner interprets clean pages as unmarked and dirty pages as marked. In figure 1d page 3 is unmarked from being marked.

Further **Printezis** discloses dirty cards are changed to precleaned, which are considered dirty by generational collection, but considered clean in the final mark phase (**Printezis** Page 17, 4.9 Concurrent Precleaning) and a young generation collection cleans a dirty card, the information that the card has been modified must be recorded for the mostly-concurrent collector (**Printezis** Page 9, 4.2 Using the card table).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of the cited references because **Printezis** teaching would have provided a faster tracing by unmarking any marked card that does not contain at least one of said marked objects.

***Conclusion***

7. The prior art made of record and not replied upon is considered pertinent to applicant's disclosure is listed on 892 form.

***Contact Information***

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Usmaan Saeed whose telephone number is (571)272-4046. The examiner can normally be reached on M-F 8-5.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hosain Alam can be reached on (571)272-3978. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Usmaan Saeed  
Patent Examiner  
Art Unit: 2166



Leslie Wong  
Primary Examiner

US  
August 7, 2006